

**PATENT APPLICATION**

**FLEXIBLE BUFFERING SCHEME FOR MULTI-RATE SIMD  
PROCESSOR**

**Inventors:**

Fred Stacey, a citizen of Canada, residing at  
36 Sarah Street, Carleton Place, Ontario K7C 2Z5 Canada

Christian Bourget, a citizen of Canada, residing at,  
97 Meridien, Hull, Quebec J9A 3C6 Canada

**Assignee:**

Catena Networks, Inc.  
303 Twin Dolphin Drive, Suite 600  
Redwood Shores, CA 94065

**Entity:** Small business concern

## **FLEXIBLE BUFFERING SCHEME FOR MULTI-RATE SIMD PROCESSOR**

The present invention relates to single instruction, multiple data (SIMD) processors. In particular, the present invention relates to rate buffering in digital subscriber line (DSL) systems.

### **BACKGROUND OF THE INVENTION**

A single instruction, multiple data (SIMD) processor essentially consists of a single program control unit (PCU) that controls the data processing of any number of data channels. FIG. 1A illustrates one possible SIMD configuration, represented by the numeral 10. The PCU 12 is coupled to N data paths 13, implying a parallel-processing scheme. Each data path 13 is coupled to a data memory 14 and a processor 15.

FIG. 1B illustrates a second SIMD configuration, represented by the numeral 16. In this configuration, the sharing of the instruction is done serially by time division multiplexing the processing in a data path 18 for all channels. The data is, therefore, changed N times at the input to the data processor 15 for every instruction that is produced.

The advantages that the SIMD architecture provides are savings in both power consumption and area reduction when compared to a solution using dedicated processors for each data path, or channel. The savings come from having only one PCU 12 and one program memory. If the data path processor is also being time shared, as in FIG. 1B, further savings in area reduction are realized. However, the processor must be able run at higher speeds to accommodate multiple channels. Furthermore, simplifications are also made at a higher level, since there is only one program load to manage and track. Having only one program load reduces start-up times if download times are consuming a large portion of time.

As described in the various standards defining the different flavors of digital subscriber line (DSL) systems, the procedure from power-up to run time operation takes modems through a series of handshakes and initialization procedures. These procedures require the modems to handle different data rates while maintaining a constant carrier frequency profile. In a multiple channel system, the assumption is that all channels may not be in the same state at any given time.

The maintenance of the constant carrier frequencies facilitates reuse of program code to perform some of the necessary tasks such as fast Fourier transforms (FFTs),

equalization and the like. However, the changing data rates make it difficult to use one processor for performing symbol-based operations on multiple channels. This is due to the fact that the modem cannot synchronize all channels with its own processing rate since the symbol rate for all channels is not equal. Therefore, the presence of different rates in a multi  
5 channel system precludes using a constant rate processor for all channels.

It is an object of the present invention to obviate or mitigate some of the above disadvantages.

### BRIEF SUMMARY OF THE INVENTION

10 In accordance with the present invention, there is provided a single instruction multiple data (SIMD) architecture for controlling the processing of plurality of data streams. The SIMD architecture comprises a memory for storing the data from the channels, a processor operatively coupled with the memory for processing data from the data streams, and a controller for controlling the processor. Storing the data in the memory de-couples the  
15 operating rate of the processor and the operating rate of the data streams.

In accordance with a further aspect of the present invention, there is provided a method for controlling the processing of multiple data streams in a SIMD architecture. The method comprises the steps of storing data in a memory as it is received; determining, at predetermined times, whether all of said data has been received; providing a signal indicating  
20 that all of the data has been received; using the signal to determine which data to process; and processing the data.

### BRIEF DESCRIPTION OF THE DRAWINGS

25 An embodiment of the invention will now be described by way of example only with reference to the following drawings, in which:

FIG. 1A is a block diagram of a standard SIMD architecture with multiple data paths;

FIG. 1B is a block diagram of a standard SIMD architecture with a single, time-shared data path;

30 FIG. 2 is a block diagram of a circular buffer architecture; and

FIG. 3 is a block diagram of a SIMD architecture according to an embodiment of the invention.

## DETAILED DESCRIPTION OF THE INVENTION

For convenience, in the following description like numerals refer to like structures in the drawings.

FIG. 2 illustrates a circular buffer architecture, represented generally by the numeral 20. The circular buffer 20 is partitioned into three distinct sections. The first section 22 is for pre-processed symbols, the second section 24 is for present symbol processing, and the third section 26 is for post-processed symbol extraction. A symbol manager 28 is used for managing the locations of these symbols.

The buffer 20 may include an elastic region that is able to absorb data growth or depletion due to differences in rates of the three devices (output device, input device, and processor) that use the buffer 20. This region may hold up to one symbol, and may be located within the first section 22.

FIG. 3 illustrates a SIMD architecture, represented by the numeral 36. The architecture 36 includes a PCU 12, multiple data paths 13, multiple data memories 14 and multiple processors 15. The architecture also includes enable signals 32, coupled to the processors 15.

Referring to FIG. 2, data is typically input serially into the pre-processed section 22. Once the data has been received, it is rotated to the present symbol processing section 24, where it is parallel-processed. Once the processing is complete, the symbol is rotated to the post-processed section 26 of the buffer 20, where it is output serially. Although the symbol is rotated through several sections of the buffer 20, its physical location does not necessarily change. Changing the location of the symbol can be done; however, it would require more time and more memory.

Maintaining the same location for a particular symbol is accomplished since the buffer 20 is circular. Rather than have the address of the symbol physically rotate, the sections 22, 24, and 26 of the buffer 20 rotate about predetermined addresses. Therefore, an address that points to an incoming symbol is in the pre-processed section 22. Once the symbol has completely arrived and is being processed, the address that points to that symbol is in the processing section 24. Once the symbol has been processed, that address is considered to be in the post-processed section 26.

The symbol manager 28 locates the base address for each of the symbols, allowing the circular nature of the buffer 20 to be transparent to each device accessing the data. The input data enters the buffer 20 at an arbitrary data rate. The data is loaded

sequentially into the pre-processed section 22 until a complete symbol is collected. At that point, the symbol manager 28 advances to the next base pointer location.

(As an added feature, the address generation unit can access the buffer 20 directly with the address offset from the processor without the addition of the base address from the symbol manager 28, by way of a switch. This allows the processor 15 to bypass the symbol manager 28 and access the buffer 28 absolutely.)

The PCU 12 indicates the start of a processing cycle with a start of processing (SOP) pulse. At each SOP pulse, the base pointer for the processing section 24 is compared to the base pointer for the incoming symbol (in the pre-processed section 22). The difference between these base pointers indicates whether or not a full symbol is ready for processing. If a full symbol is present, the enable signal 32 (shown in FIG. 3) for that symbol is activated. Otherwise, the enable signal 32 remains inactive and the comparison is done again at the next SOP. Therefore, only the processors 15 that have received a complete symbol are enabled.

As each of the devices completes processing its respective symbol, the symbol manager 28 advances the base pointer of the processing section 24 to the next symbol. Once the base pointer of the processing section 24 advances, the processed symbol is in the post-processed section 26. The extraction of the post-processed data is slaved to the processor 15, and is only performed after the symbol has been processed.

An advantage of this type of buffering scheme is that the processor is decoupled from the incoming data rate of each channel. This is true with the restriction that the SOP of the processor is greater than or equal to the maximum baud rate of the channels. If this were not true, it is possible that incoming data could overwrite previously received data before it is processed. Therefore, the net processing rate of each channel is approximately equal to the baud rate for that channel since its processor 15 may be periodically disabled.

The rate at which any given channel is disabled (assuming zero jitter between each of the baud rates) is given by:

$$\%PROC_{OFF} = \frac{F_{baud\_SOP} - F_{baud\_CHAN}}{F_{baud\_SOP}}$$

This equation also indicates the "bursty" nature of the data output rate. That is, the output is provided in bursts -- when the processor is enabled -- rather than a constant steady stream. Also, the varying instantaneous latency due to the gapped processing can be determined.

Since the data is assumed to be arriving at a constant input rate, any gaps in the processing increase buffering requirements. However, since the worst case, or fastest,

baud rate of the channel is equal to the baud rate of the processor, the buffering requirement is limited to the symbol size for each of the three sections 22, 24, and 26.

Implementing an SIMD in this manner provides several advantages. The architecture ultimately results in a net decrease in gate count and expended power, since the processors are only used for completely received symbols. Buffering requirements can be combined with those necessary for other considerations in the signal processing. Therefore, little or no extra memory is required. The structure can be applied to any symbol size. This includes processing on a sample by sample basis. The structure can be expanded to accommodate any number of channels. Lastly, this structure has direct applications to implementations of ITU G.992.2 (and other standards) for DSL systems, since the baud rate changes throughout operation.

In an alternate embodiment, it is possible that the data is received in parallel and the output transmitted in parallel.

In yet another embodiment, it is possible that the data is received serially and the output transmitted in parallel.

In yet another embodiment, it is possible that the data is received in parallel and the output transmitted serially.

It is possible to implement the system as described above using other SIMD implementations and will be apparent to a person skilled in the art.

Although the invention has been described with reference to certain specific embodiments, various modifications thereof will be apparent to those skilled in the art without departing from the spirit and scope of the invention as outlined in the claims appended hereto.